Brazilian Houses

Francesco C., Giulia G., Gaia N., Gabriele P.

2023-06-12

1.Picking and Understanding a Dataset

We opted for the Brazilian houses dataset because it motivates us in investigating the driving forces behind high rents, in this case in Brazil, and it recreates a real case scenario where we as analysts must inform the new company's entry into the real estate market.

Our goal is to build a predictive model that determines the rental amount based on specific characteristics of houses. This will provide valuable insights into the housing rental market in major cities across Brazil.

We first load libraries that will be useful.

To start our project, we will investigate some trends and important aspects of the Brazilian's house-rent market. We start by importing the dataset and having a little overview of it.

Loading the dataset

```
#The read.csv function in R is used to read data from a CSV (Comma-Separated Values) file
#and store it as a data frame in R.
Data <- read.csv('BrazHousesRent.csv')

#little OVERVIEW of the dataset: structure and summary

#The str() function in R is used to display the structure of an R object.
#It provides a concise summary of the object's internal structure, including its type,
#dimensions, and the data contained within it.
str(Data)

## 'data.frame':    10692 obs. of  12 variables:
##  $ city                : chr  "São Paulo" "São Paulo" "Porto Alegre" "Porto Alegre" ...
##  $ area                : int  70 320 80 51 25 376 72 213 152 35 ...
##  $ rooms               : int  2 4 1 2 1 3 2 4 2 1 ...
##  $ bathroom            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ parking.spaces      : int  1 0 2 0 0 2 0 2 1 0 ...
##  $ floor               : chr  "7" "20" "6" "2" ...
##  $ animal              : chr  "acept" "acept" "acept" "acept" ...
##  $ furniture           : chr  "furnished" "not furnished" "not furnished" "not 
furnished" ...
##  $ hoa..R..            : int  2065 1200 1000 270 0 0 740 2254 1000 590 ...
##  $ rent.amount..R..    : int  3300 4960 2800 1112 800 8000 1900 3223 15000 2300 ...
##  $ property.tax..R..   : int  211 1750 0 22 25 834 85 1735 250 35 ...
##  $ fire.insurance..R..: int  42 63 41 17 11 121 25 41 191 30 ...
```

The structure analysis provides a clear understanding of the data's characteristics. The data frame consists of four factors, representing categorical variables that store data as levels. These factors can accommodate both strings and integers.

Specifically, "floor," "city," "animal," and "furniture" are factors, while the remaining eight variables are integers.

```
#The summary() function is widely used for analyzing data frames, where it provides
summaries for each variable in the data frame.
#For numeric variables, it displays the minimum, 1st quartile, median, mean, 3rd
quartile, and maximum values.
#For factors and other categorical variables, it presents the frequency counts of each
level.
#summary(Data)
```

From the summary study we can notice something intriguing: the variable "hoa..R.."(Monthly Homeowners Association Tax), ranges from 0 to 1117000, but has a mean of 1174, this means that the small values and the extremely big values are not many. This gives us a hint on the presence of outliers, which presence is not surprising since the dataset is created by a web crawler.

Same thing is visible for other variables such as: - "property.tax..R.." (Yearly property taxes in Real), ranges from 0.0 to 313700.0 but the 1st quartile, Median and 3rd quartile are lower that 400 and the mean is 366.7, so the Maximum is way higher than all these observations, clearly it is an outlier. - "area" (Property area), ranges from 11.0 and 46335.0, same reasoning as the other variables, probably all the big values are few,since the mean has a quite smaller value (149.2).

By analyzing the structure of the data we can see that floor is stored as character but should be a number (since it expresses the floor number of the house), so we must convert it. We can also see that animal, city and furniture could be better expressed as factors.

Cleaning dataset

We then check for missing values. NA stands for "Not Available", they are special values that represents missing data. It is important to know how many NA values are present in the dataframe.

```
##                city                area               rooms            bathroom
##                   0                   0                   0                   0
##      parking.spaces               floor              animal           furniture
##                   0                2461                   0                   0
##            hoa..R..    rent.amount..R..   property.tax..R.. fire.insurance..R..
##                   0                   0                   0                   0
```

We can see that only floor contains null values, we will deal with them later!

We then proceed by dropping any duplicates.

```
#dropping DUPLICATES
Data <- Data[!duplicated(Data),] #dataframe with no duplicates
```

Another annoying aspect we noticed from analyzing the structure of the dataset is the name of some variables, so we changed it. And since it is convenient we also move the target variable (rent.amount) as last column in the dataset.

```
##Let's move the target(rent) variable to the last column
Data <- Data[, c(names(Data)[names(Data) != "rent.amount..R.."], "rent.amount..R..")]

#lets CHANGE NAMES to the columns for the sake of simplicity
Data <- rename(Data, monthly.tax = hoa..R..)
Data <- rename(Data, rent.amount = rent.amount..R..)
Data <- rename(Data, property.tax = property.tax..R..)
Data <- rename(Data, fire.insurance = fire.insurance..R..)
```

Dealing with NULL Values

Let's deal with the null values in the floor column :

The interpretation of floor's values is that floor 1 represents the ground floor, as it works is a lot of states, so we decided to use mean or median imputation for replacing the nulls in the floor column. In particular we will use PMM Mice method, from the MICE Package, which stands for predictive mean matching, a regression model which uses the other variables as predictors; trivially, it uses rows having similar predictors of the missing value row, and impute the missing value of the row by using the "similar" values as predictors.

The code is reported in Rscript

After all these cleaning and changing is useful to see a summary of our "new" data:

```
##               city           area              rooms          bathroom
##  Belo Horizonte:1209   Min.   :   11.0   Min.   : 1.00   Min.   :1.000
##  Campinas      : 823   1st Qu.:   59.0   1st Qu.: 2.00   1st Qu.:1.000
##  Porto Alegre  :1154   Median :   95.0   Median : 3.00   Median :1.000
##  Rio de Janeiro:1431   Mean   :  152.5   Mean   : 2.54   Mean   :1.283
##  São Paulo     :5712   3rd Qu.:  190.0   3rd Qu.: 3.00   3rd Qu.:1.000
##                        Max.   :46335.0   Max.   :13.00   Max.   :9.000
##  parking.spaces     floor                 animal           furniture
##  Min.   :0.00   Min.   :  1.000   acept    :8073   furnished    :2515
##  1st Qu.:1.00   1st Qu.:  3.000   not acept:2256   not furnished:7814
##  Median :2.00   Median :  5.000
##  Mean   :1.33   Mean   :  6.668
##  3rd Qu.:2.00   3rd Qu.:  9.000
##  Max.   :8.00   Max.   :301.000
##   monthly.tax       property.tax      fire.insurance    rent.amount
##  Min.   :      0   Min.   :     0.0   Min.   :  3.00   Min.   :  450
##  1st Qu.:    180   1st Qu.:    41.0   1st Qu.: 21.00   1st Qu.: 1599
##  Median :    571   Median :   130.0   Median : 37.00   Median : 2750
##  Mean   :   1092   Mean   :   377.1   Mean   : 54.28   Mean   : 3967
##  3rd Qu.:   1289   3rd Qu.:   390.0   3rd Qu.: 70.00   3rd Qu.: 5000
##  Max.   :1117000   Max.   :313700.0   Max.   :677.00   Max.   :45000
```

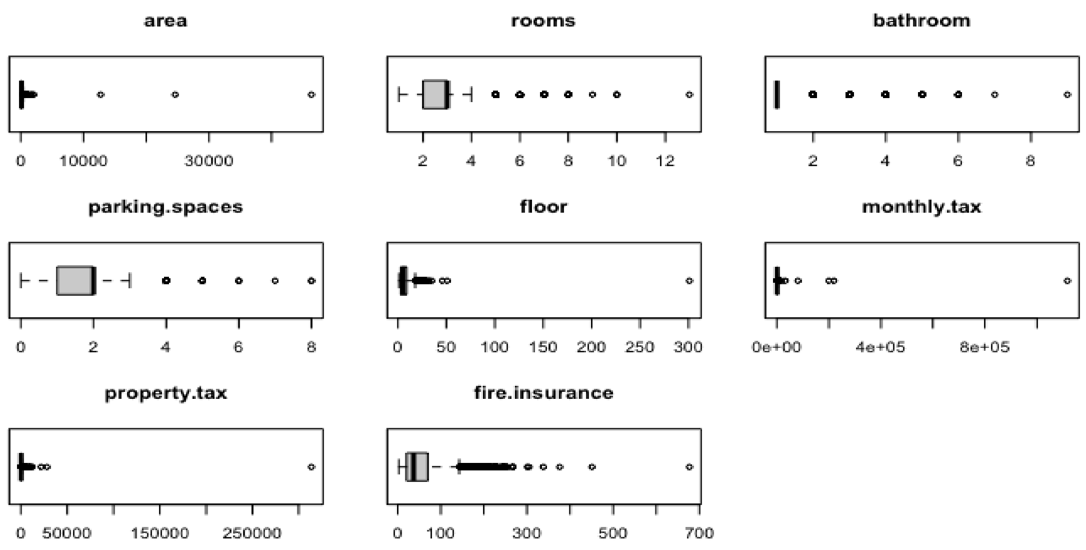
We must keep in mind,as previously announced, that the data have been gathered through a web-crawler so it is crucial to check and eliminate outliers, as they have the potential to disrupt the integrity of our results and the accuracy of our models.

Outliers

Outliers are data points that are significantly different from the majority of the data. They can impact statistical analyses and may indicate unique observations or data issues. Handling outliers depends on the context, and they can be removed, adjusted, or given special consideration based on their impact and underlying reasons. In our case we will remove them since most of them are probably errors (for example 301 in "floor" which probably was meant to be 30 since the highest building in Brazil has 51 floors).

An important question to answer is from which variables should we remove outliers? From the summary analysis we can notice that most of the numerical variables seem to contain outliers, since their max or/and min differ so much from mean/median, but let's plot these variables to understand better if all of them contain really outliers or not.

```r
# We select numeric features
numeric_cols <- sapply(Data, is.numeric)
numeric_features <- names(numeric_cols)[numeric_cols]
numeric_features <-  numeric_features[-9] #we exclude the target variable
```



From these boxplot we can surely state that 'area', 'monthly.tax', 'property.tax', 'fire.insurance', 'rent.amount', 'parking.spaces', 'rooms' and 'floor' contain outliers. The variable from which we will NOT remove outliers with the interquartile method is "bathroom", since we have noticed that there are too many houses that have only one bathroom, considering it in the removal of outliers, would mean removing too many rows.

Remove OUTLIERS using interquartiles

```
#looking for outliers
outlier_cols <- c('area',
'monthly.tax','property.tax','fire.insurance','rent.amount','parking.spaces','rooms','flo
or')
brz_col <- brz_data[, outlier_cols]
q1 <- apply(brz_col, 2, quantile, probs = 0.25, na.rm = TRUE)
q3 <- apply(brz_col, 2, quantile, probs = 0.75, na.rm = TRUE)
iqr <- q3 - q1
upper <- q3 + 1.5 * iqr
lower <- q1 - 1.5 * iqr
outliers <- apply(brz_col, 1, function(x) any(x < lower | x > upper, na.rm = TRUE))

# Identify outliers rows
outlier_rows <- row.names(brz_col)[outliers] # Print the number of outliers detected

# Drop rows with outliers
#Cleaned Data is our new dataframe containing all independent variables encoded and
without outliers
brz_cleaned <- brz_data[!outliers,]
```

The outliers detected are 2079 which is a pretty good number considering the dimension of our dataset and its origin.

2. Describing the data: EDA

Before exploring all the relations between variables is important to distinguish the type of features we are dealing with:

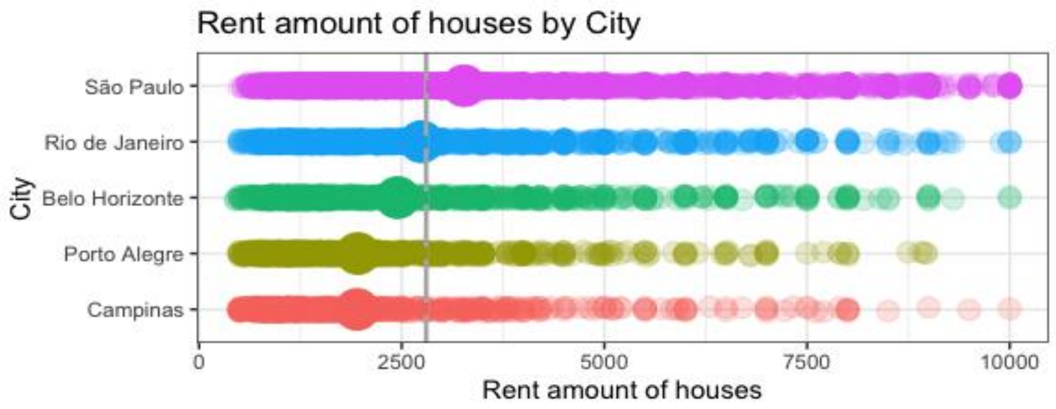
- rent.amount which is our response variable
- discrete: (floor,parking.spaces,rooms,bathroom)
- continuous: (area, montly.tax, property.tax, fire.insurance)
- categorical: (animal,city,furniture)

We will proceed providing a visual description of the ones which can be more explicative for our analyses.

Let's start by taking into account Categorical Variables. Since this data have been collected in order to better understand the house-rent market in some of the most important cities in Brazil, the geographical positioning of the houses is a main feature to take into consideration for the analysis.

CATEGORICAL VARIABLES

Let's start by analyzing how the rent amount changes according in which city the house is located.



The grey line represents the average rent for all houses. The small dots represent all the observations (houses) for each city, while the big dots represent the average rent amount for each specific city. We can see that the house located in San Paolo have a rent amount above average, the houses located in Rio de Janeiro have a rent amount close to average and the ones located in Belo Horizonte have a rent amount slightly less than average and the houses in Campinas and Porto Alegre have a rent amount less than average.

We will use a barplot to study the city, animal, furniture variables and to better answer to these questions:

- which is the city where it is easier to find a house to rent?
- Are pets usually allowed when renting a house in Brazil?
- If I want to rent a house in Brazil should I worry about possibly having to pay for my furniture?

We decided not to show the barplots but only to comment them

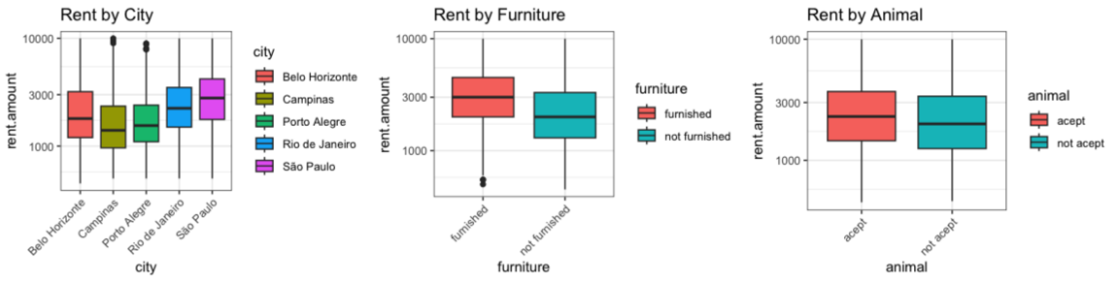
From these graphs we can notice that:

Most houses (to rent) are located in San Paolo, this makes sense since it is the biggest city of Brazil. The city with less frequency, so less houses, according to the dataset, is Campinas.

It is also evident that the houses that accept animals are more with respect to the houses that do not accept them.

Lastly we can observe that most houses are not furnished, this makes sense since we are studying a rent-house dataframe, usually when you rent a house, you have to provide yourself with furniture.

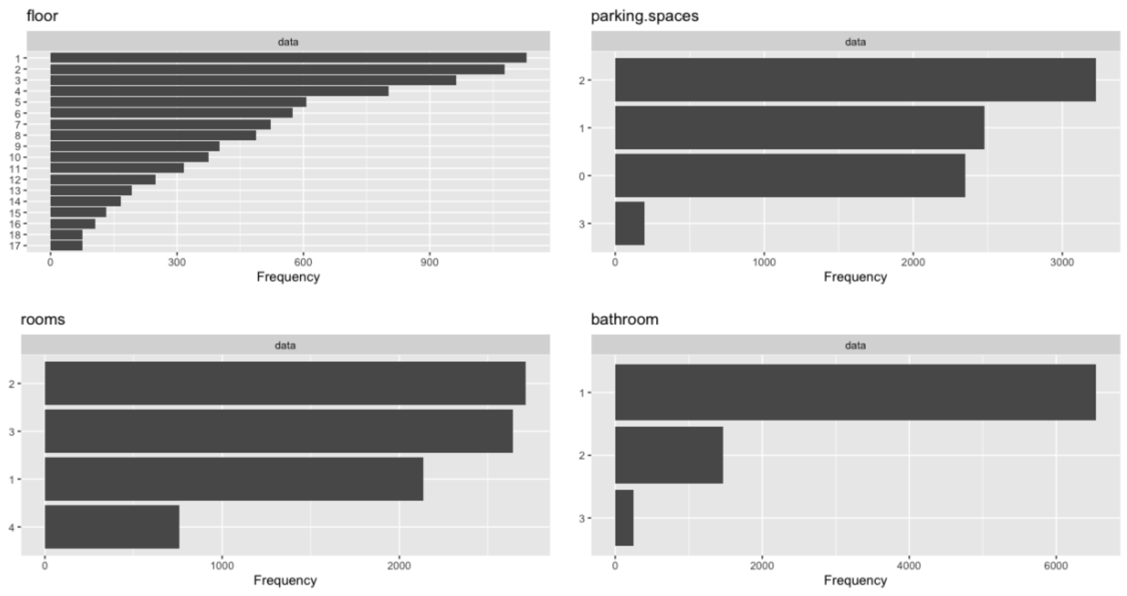
Let's explore even better how RENT.AMOUNT is affected by the categorical variables, through a boxplot.



From this boxplots we can see that houses in Sao Paulo cost more than in other places as the interquartile and the median show (coherent with our first plot). Also, the majority of the most expensive are located in this city. While pet allowance does not make a huge difference in rent prices, furniture does. The lower quartile of the furnished home rental price corresponds to the median of not furnished ones, indicating a relevant difference in rental prices.

NUMERICAL VARIABLES: DISCRETE

Let’s now take into consideration discrete variables how are the values distributed for each feature. Let’s use a barplot:



From the graphs we can say that:

- most houses have only 1 bathroom, there are just a few that have more than 3.
- the room graph shows us that very few houses have 7 rooms, while the majority have 2/3.
- from the parking.space bar plot we can see that many houses have no parking space, but there are a lot that have at least 1 or 2.
- the floor graph shows us that in Brazil it is more frequent to rent houses at lower floors.

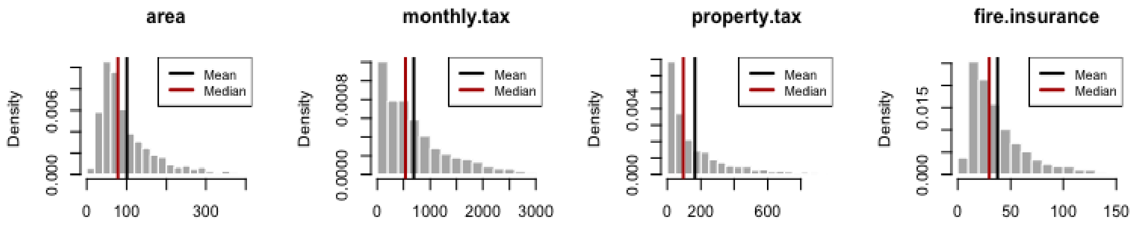
Now let’s see how this variables relate with the target variable rent.

We decided not to show the plots but only to comment them.

Observing these boxplot we can state that they are more or less all skewed, meaning that higher values (of floor, room, parking space, bathrooms) correspond to higher rent amount, this makes absolutely sense! For example we can notice that the houses having 4 or more rooms have higher rents amounts. We notice that houses with 5 bathrooms have higher rent amounts. For parking space we can see that the more parking space the more the rent amount this is quite intuitive.

NUMERICAL VARIABLES: CONTINOUS

We will use histogram to describe the distribution of the continuos variables. A histogram provides a graphical representation of the data by dividing it into intervals called “bins” along the x-axis and displaying the frequency or count of observations within each bin on the y-axis.



Observing these histogram we can state that all the continous variables are right skewed, meaning that the majority of the data points are concentrated towards the left side of the histogram, while a smaller number of data points have higher values that extend towards the right. We can see in this histogram that the mean is larger than the median since the mean is influenced by the extreme values in the right tail. The median,instead, being a robust measure of central tendency, is less affected by extreme values and is closer to the bulk of the data. This is why we must scale our data.

Scaling Data

Scaling data is important because:

It helps avoid bias by ensuring that all features contribute equally. Scaling enhances convergence, making optimization algorithms more efficient. It improves model performance, especially for distance-based algorithms. Scaling facilitates the interpretation of feature importance. Certain algorithms require standardized or normalized data for accurate results.

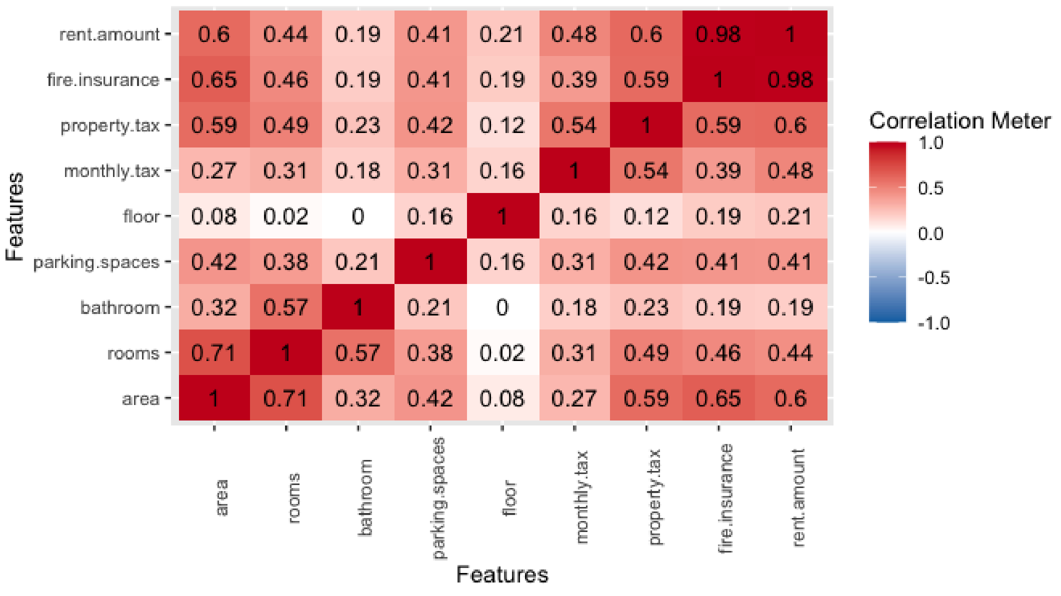
It is not necessary to scale discrete variables since they already have a predefined set of distinct values. Scaling is typically applied to continuous variables to ensure that they are on a similar scale and to prevent any potential bias in the model. It is why we are not scaling the variables *rooms*, *bathroom*, *parking.spaces*, *floor*.

```
variables_to_scale <- c("area", "monthly.tax", "property.tax", "fire.insurance")
scaled_var <- scale(brz_cleaned[, variables_to_scale])
finaldata <- brz_cleaned
finaldata[, variables_to_scale] <- scaled_var
```

For our regression task we will use the new version of the dataset: finaldata

Before analyzing how the continuous variables interact with the target (rent.amount), let’s plot the correlation matrix:

CORRELATION MATRIX



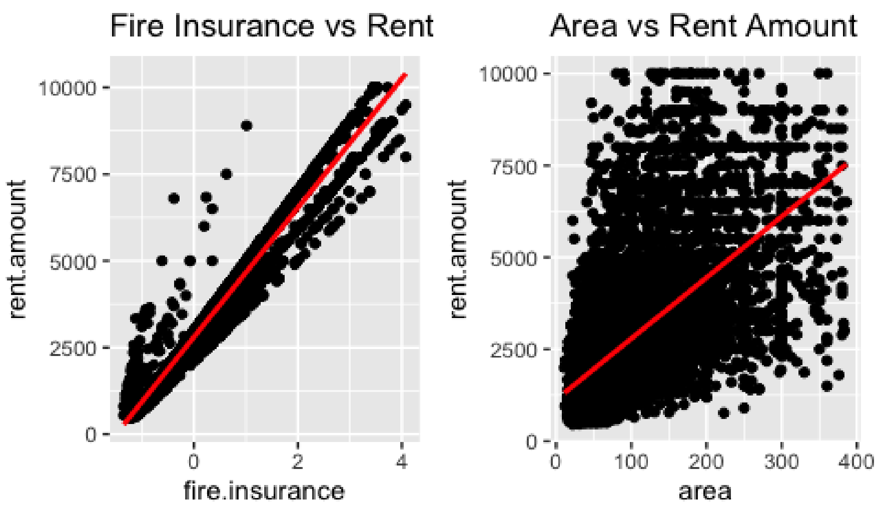
Interesting insights from the correlation matrix:

- remarkable correlation between the target variable and fire.insurance;
- rooms,area and rent.amount are correlated and is quite intuitive why: one would want to spend more if there are more rooms and space in a house and viceversa.
- since the only 2 highly correlated variables are the fire insurance and the target rent.amount there is no apparent risk of multicollinearity among predictors.

So the most interesting relationship to inquire are the one between:

- area vs rent.amount
- fire.insurance vs rent.amount

To see how this variables are related with the response variable we will use scatterplots.



Between the two scatterplot the most striking is the one between fire.insurance and rent.amount, that show that even taking the fire.insurance as a single predictor could be a decent solution,given the high positive correlation. Also area seem to have a positive relation with rental prices.

3. Objective of the first task

Our primary goal was initially to identify the features that would make real estate more profitable. This objective holds practical significance for real estate companies, property owners, renters, and market analysts. It provides valuable insights into rent pricing, facilitates decision-making processes, and enhances overall understanding of the dynamics within the rental market. Thanks to the EDA we've already observed how rental prices are higher in some areas (Sao Paulo) and how some characteristics significantly increase the rent price (higher number of parking spaces, bathroom, rooms, floor) and we've highlighted that there is a strong correlation between fire insurance and rent.amount.

We want to be able to answer questions like:

-Are there significant price variations among different cities?

-What would be a fair price for rent if the house has 5 rooms and 2 bathrooms or if the house is located in this area?

-If I have a pet it more convenient to search a house in which area?

4. Lower-dimensional model

Splitting the dataset

We use the dataset finaldata, as is the one already scaled and encoded.

75% of data will be for the train set, and 25% for the test.

```
set.seed(123)
idx <- sample.int(n = nrow(finaldata), size = floor(.75*nrow(finaldata)), replace = F)

test_set <- finaldata[-idx,]

x <- finaldata[,1:11]
y <- finaldata$rent.amount

# Training set
x_train <- x[idx, ]
y_train <- y[idx]

# Test set
x_test <- x[-idx, ]
y_test <- y[-idx]

brz_train_sc <- x_train
brz_train_sc$rent.amount <- y_train
```

After seeing the high correlation between fire.insurance and rent.amount and plotted their relation, we need to build a linear regression with fire.insurance as only predictor.

Linear Regression with 'fire.insurance'

Linear regression is a statistical method utilized to determine a linear association between a dependent variable and one or more independent variables. Its objective is to identify the line of best fit that minimizes the disparity between observed and predicted values. In linear regression, the p-value plays a crucial role as it assesses the null hypothesis that the coefficient for an independent variable is zero, indicating no relationship.

```
##
## Call:
## lm(formula = rent.amount ~ fire.insurance, data = brz_train_sc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2413.6  -155.3    17.1   131.3  4204.6
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2806.019      4.375   641.4   <2e-16 ***
## fire.insurance 1867.774      4.402   424.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 344.1 on 6185 degrees of freedom
## Multiple R-squared:  0.9668, Adjusted R-squared:  0.9668
## F-statistic: 1.8e+05 on 1 and 6185 DF,  p-value: < 2.2e-16

## RMSE with linear model on test set: 353.5419
```

The R-squared value of 0.9668 indicates that approximately 96.68% of the variance in the rent amount can be explained by the linear regression model with fire insurance as the only predictor, suggesting that the fire insurance variable, has a strong ability to explain the variation in rent amounts.

The adjusted R-squared value of 0.9668 takes into account the number of predictors in the model and penalizes the addition of unnecessary variables. Since there is only one predictor in this model, the adjusted R-squared value is the same as the R-squared value. This implies that the inclusion of additional predictors would not significantly improve the model's explanatory power.

The RMSE, 353.5419, for the linear model with only "fire.insurance" is relatively low compared to the range of the response variable.

Linear Regression with 'area'

Since we have noticed more than once the importance of fire.insurance as predictor, we think is interesting to see a linear regression model which has as predictor area instead of fire insurance, as area is definitely less correlated to rent amount.

```
##
## Call:
## lm(formula = rent.amount ~ area, data = brz_train_sc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5592.8  -962.1  -382.8   594.4  7541.4
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2802.45      19.28  145.34   <2e-16 ***
## area         1133.98      19.45   58.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1517 on 6185 degrees of freedom
## Multiple R-squared:  0.3548, Adjusted R-squared:  0.3547
## F-statistic:  3401 on 1 and 6185 DF,  p-value: < 2.2e-16

## RMSE with linear model on test set: 1527.402
```

The multiple R-squared value of 0.3548 indicates that approximately 35.48% of the variance in the rent amount is accounted for by the area. The adjusted R-squared in this case is 0.3547, which is slightly lower than the multiple R-squared. The RMSE is 1527.402, indicates the average prediction error in terms of rent amount. The results suggest that a model using only the area explains a moderate amount of the variation in rent amount. However, there is still a considerable amount of unexplained variability, as indicated by the RMSE value. It is important to consider other factors that influence the rent amount to improve the model's performance.

It's worth to notice that as we remove fire.insurance as predictor the RMSE falls considerably.

We build a multiple linear regression for our lower dimensional model.

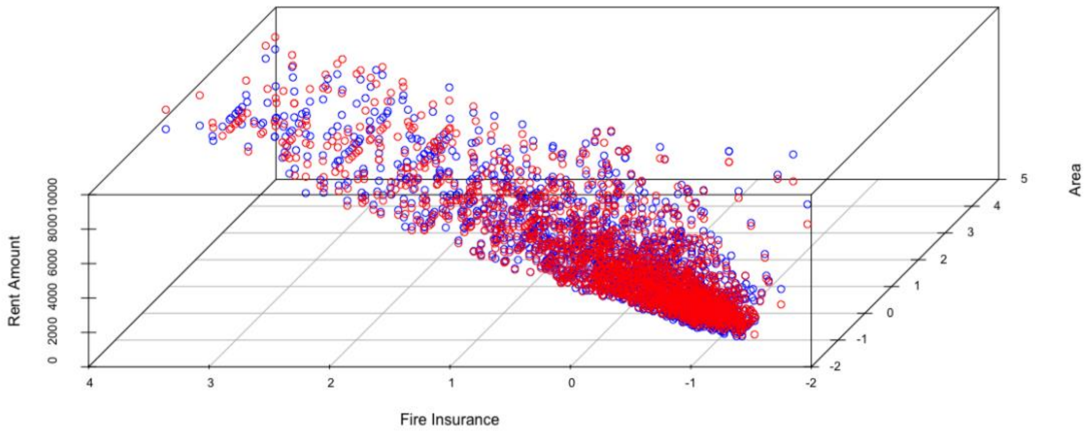
Multiple regression: fire.insurance + area

Let's try adding to the linear model another predictor. We decided to insert area, because in our opinion it partly encloses the information of variables like parking.space, room, bathroom,and floor and it has a trivial positive correlation with rent.amount.

Multiple regression is a statistical technique used to model the relationship between a dependent variable and two or more independent variables. It extends simple linear regression, which involves only one independent variable, to incorporate multiple predictors.

```r
#We start building the model with predictors: area and fire.insurance
multiple_linear_model <- lm(rent.amount ~ fire.insurance + area , data = brz_train_sc)
#summary(multiple_linear_model)
```

The p-value in linear regression tests the null hypothesis that the coefficient for an independent variable is zero (no relationship). As in this case, a p-value lower than 0.05 suggests a statistically significant relationship between the independent variables and the dependent variable. The R-squared quantifies the proportion of variance explained by the independent variables, with higher values indicating a better fit. In our case the Adjusted R-squared is high (0.9693),so the variance explained is big.



The plot consists of three axes: the x-axis represents the fire insurance , the y-axis represents the rent amount, and the z-axis represents the area. Each *blue* data point corresponds to the observations (combination of area, fire insurance, and the actual rent amount), while *red* points on the plot represent the predicted rent amount based on the multiple linear regression model (which has fire.insurance and area as predictors) using the test data.

We know proceed measuring the performance of our models on the train and on the test set. We will use R^2,Adjusted R^2 and RMSE.

R-squared (coefficient of determination) measures the proportion of the variance in the dependent variable that can be explained by the independent variables in the model. It ranges from 0 to 1, with higher values indicating a better fit.

Adjusted R-squared is a modification of R-squared that penalizes the inclusion of irrelevant variables in the model.

```r
# Calculate R-squared
r_squared <- summary(multiple_linear_model)$r.squared

# Calculate adjusted R-squared
n <- length(finaldata$rent.amount)
p <- length(coef(multiple_linear_model)) - 1  # Number of predictors (excluding
intercept)
adjusted_r_squared <- 1 - (1 - r_squared) * ((n - 1) / (n - p - 1))

## R-squared: 0.9692678

## Adjusted R-squared: 0.9692604
```

The R-squared value of 0.9692678 indicates that approximately 97% of the variance in the rent amount can be explained by the independent variables (area and fire insurance) included in the model. This is a high R-squared value, suggesting that the model is able to explain a significant amount of the variation in the rent amount based on these two variables.

The adjusted R-squared value of 0.9692604 takes into account the number of predictors in the model and penalizes the addition of unnecessary variables. It is very close to the R-squared value, indicating that the inclusion of area and fire insurance as predictors has not resulted in a significant reduction in the adjusted R-squared value. This suggests that these two variables are highly relevant in explaining the rent amount and that the model is not overfitting.

RMSE (Root Mean Squared Error) is a measure of the average prediction error of a regression model. It represents the square root of the mean squared differences between the predicted and actual values. A lower RMSE indicates a better fit of the model, with smaller average prediction errors.

First let's calculate RMSE on the training set:

We now predict on the test set and measure the actual performance of our multiple regression.

```
## RMSE on the test data: 342.3592
```

Comparing the RMSE of different models helps identify the one that provides more accurate predictions for the dependent variable. The RMSE, 342.3592, for the multiple regression model with only "fire.insurance" and "area" is relatively low compared to the range of the response variable.

Comparing the RMSE values, we can observe that the RMSE with both the fire insurance and area variables as predictors (342.36) is slightly better than the RMSE when using only the fire insurance variable as a predictor (353.54). Therefore, based on the RMSE values alone, it appears that incorporating the area variable in addition to the fire insurance variable improves the model's predictive performance and leads to more accurate rent amount predictions. Although is important to point out that adding "area" did not make an enourmous difference on the RMSE.

Let's try to build more complex models.

5. Complex models

Multiple linear regression model using all predictors

We implement a multiple regression to predict rent.amount using all features as predictors.

```
## R-Squared with multiple regression:  0.9821303

##
## RMSE (train set) with multiple regression: 252.3383

##
## RMSE (test set) with multiple regression: 254.8276
```

The R-squared value of 0.9821303 indicates that approximately 98.21% of the variance in the rent amount can be explained by the multiple regression model that includes all the features as predictors.

We can see that using all features as predictors has improved our RMSE to 254.83 on the test set.

AIC

The Akaike Information Criterion (AIC) is a statistical measure used for model selection. It balances the goodness of fit of a model with its complexity. The AIC takes into account both the model's ability to explain the data and the number of parameters or predictors used. A lower AIC value indicates a better balance between model fit and simplicity. The AIC helps in selecting a model that provides a good fit to the data while avoiding excessive complexity.

```
## RMSE with AIC: 254.7283
```

The process starts with a simple model with only an intercept (step: AIC=119157.9, brz_train_sc$rent.amount~1), then it begins adding variables based on the decrease in AIC. Each variable that significantly decreases the AIC is added to the model, and the process continues until there are no more variables that significantly decrease the AIC.

According to the final results, the most significant variables for predicting "rent.amount" in the "brz_train_sc" dataset appear to be "fire.insurance", "monthly.tax", "area", "city", "furniture", "rooms", "floor", "parking.spaces", and "bathroom", as adding these variables led to a significant decrease in AIC.

With the AIC model we achieve a RMSE of 254.7283, which suggest that the model's predictions are reasonably accurate.

Elastic Net

Elastic Net is a regularization technique that combines Lasso (L1 regularization) and Ridge (L2 regularization) penalties. It addresses the limitations of individual regularization methods and provides a balance between feature selection and parameter shrinkage in linear regression models. Elastic Net has two hyperparameters, alpha and lambda, which control the balance and strength of the regularization, respectively.

```
## RMSE (train set) with elastic net: 252.2302

## RMSE (test set) with elastic net: 254.7048
```

The RMSE values of 252.2302 on the train set and 254.7048 on the test set suggest that the elastic net model provides reasonably accurate predictions for both the training and testing datasets.

XGBoost

XGBoost, a highly advanced gradient boosting algorithm, finds extensive application in regression tasks. It effectively optimizes a loss function using a combination of gradient descent and regularization techniques. With its exceptional predictive performance, XGBoost excels in capturing complex relationships within the data. Moreover, it enables analysis of feature importance, further enhancing its capabilities.

```
#we make x_train and x_test two matrix
set.seed(123)
x_train_matrix <- data.matrix(x_train)
x_test_matrix  <- data.matrix(x_test)
xgb_train = xgb.DMatrix(data = x_train_matrix, label = y_train)
xgb_test = xgb.DMatrix(data =x_test_matrix, label = y_test)
```

We divide the code in three segments since we don't want to show all the outputs but only the code.

```
xgb_model = xgboost(data = xgb_train, max.depth = 6, nrounds =
which.min(xgbcv$evaluation_log$test_rmse_mean), verbose = 0, objective =
"reg:squarederror")

xgb_pred = predict(xgb_model, xgb_test)
rmse_xgboost <- sqrt(mean((y_test - xgb_pred)^2))

## RMSE with XGBoost: 199.8115
```

The RMSE obtain with XGBoost (199.8115) is the better since it is the lower one.

SVM

SVM regression is a machine learning algorithm used for predicting continuous outcomes.It can handle linear and non-linear relationships by using kernel functions to map the data into higher-dimensional space. In this case we also use the bootstrapping resample method which involves randomly sampling the original dataset with replacement to create multiple bootstrap samples of the same size.

```r
set.seed(123)

ctrl = trainControl(method  = "boot",
                    number  = 10)

svm_model <- train(
  rent.amount ~ .,
  data = brz_train_sc,
  method = 'svmRadial',
  trCtrl = ctrl
)

svm_pred_cv <- predict(svm_model, newdata = x_test)


rmse_svm_cv = sqrt(mean((y_test - svm_pred_cv)^2))

## RMSE with SVM 236.1984
```

Random forest using all the predictors

```
## RMSE with classic random forest: 276.0161
```

In a Random Forest model, you can assess the importance of variables using the built-in feature importance measure. Random Forest calculates feature importance based on how much each variable contributes to the overall model's performance. The higher the importance value, the more influential the variable is for prediction. The feature importance score is calculated based on the decrease in impurity or the decrease in the overall accuracy of the model when a particular feature is included in the tree-based splits.

Random Forest based on importance

```r
# Train a Random Forest model
model <- randomForest(brz_train_sc$rent.amount ~ ., data = brz_train_sc )
importance <- data.frame(importance(model))

importance<- data.frame(Variable = row.names(importance), IncNodePurity
=importance$IncNodePurity)

# Plot variable importance
#varImpPlot(model)
```

Observing the plot we see that the variables with highest importance score are: fire.insurance + area + property.tax + monthly.tax + parking.spaces + rooms, so we will use them in the Importance Random Forest model.

```
## RMSE with importance feature selection: 263.0896
```

Using the Importance Feauture selection has improved our RMSE from 276.0161 (Classic RF) to 263.0896 (Importance Score RF).

6. Conclusions on Regression

We have implemented many different models:

Complex models:

- MULTIPLE LINEAR REGRESSION - RMSE (254.83)
- AIC - RMSE(254.7283)

- ELASTIC NET - RMSE (254.7048)
- XGBOOST - RMSE (199.8115)
- SVM - RMSE (236.1984)
- RANDOM FOREST - RMSE (276.0161)
- RANDOM FOREST with importance - RMSE (263.0896)

Simple models (low dimensional):

- LINEAR REGRESSION (using fire.insurance) - RMSE (353.5419)
- LINEAR REGRESSION (using area) - RMSE (1527.402)
- MULTIPLE REGRESSION (using fire.insurance + area) - RMSE (342.3592)

Based on the evaluation of different models, including complex and simple ones, the XGBoost model stands out as the best performer in terms of RMSE.

While the XGBoost model is a non-linear model, the linear models (SVM, AIC and Multiple Regression) and the penalized model (Elastic Net) show similar performance, but with slightly higher RMSE values. These linear models may provide reasonably accurate predictions, but they may not capture the complex non-linear relationships present in the data as effectively as XGBoost. On the other hand, the Random Forest model performed less effectively, with a higher RMSE of 276.0161. This indicates that the Random Forest model may not be well-suited for capturing the specific patterns and relationships in the dataset.

Overall, the XGBoost model strikes a good balance between complexity, interpretability, and accuracy. It outperforms other models in terms of RMSE, indicating superior prediction accuracy. Therefore, based on the evaluation results, the XGBoost model is CHOSEN as the PREFERRED MODEL for rent amount prediction in this scenario.

7. Clustering

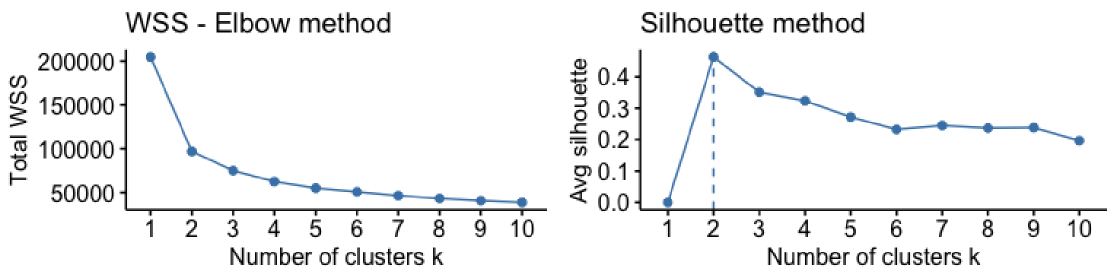
The objective of clustering is to group similar data points together based on their characteristics or features. For the clustering we want just numerical data so we use **num_v**. We scale the numerical variable with an exception for the numerical discrete variables : “bathroom”, “rooms”, “floor”, “parking.spaces”.

```r
#we scale the data
sc_num_v<- data.frame(scale(num_v[ , c('area', 'monthly.tax', 'property.tax',
'fire.insurance', 'rent.amount')]))

#add the other variables to the dataset
sc_num_v$rooms <- num_v$rooms
sc_num_v$bathroom <- num_v$bathroom
sc_num_v$parking.spaces <- num_v$parking.spaces
sc_num_v$floor <- num_v$floor
```

K-Means

Through the Elbow and Silhouette methods we do the evaluate the optimal number of cluster.

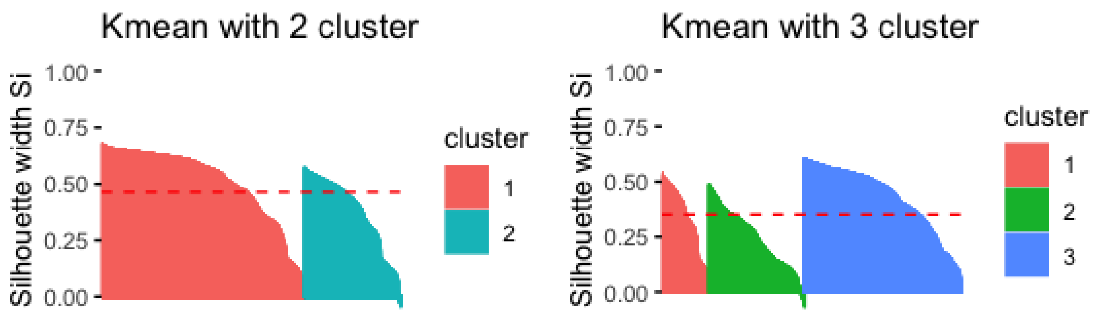
The Elbow Method



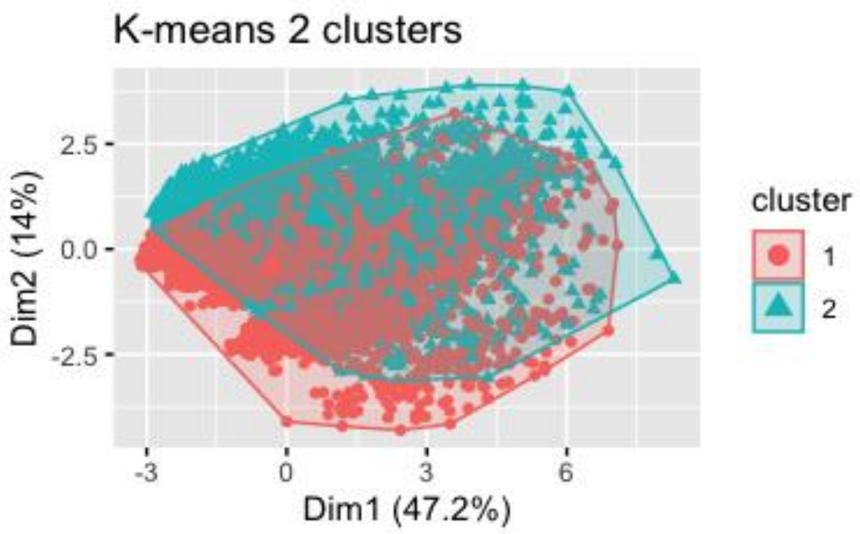
Looking at the plots from the elbow method we can see that the perfect number of clusters are 2 and the same for the silhouette method.

But to be sure we considered also 3 clusters and we proved that 2 is the right one

Through the silhouette score, that measure how well each data point fits its assigned cluster compared to other cluster, we verify the statement above.



We can see from the representation, the data are unbalance inside the cluster. As we can notice from the silhouette score > (0.50), *which suggest that the data points are relatively well-clustered*, 2 clusters are definitely the number that best fit our data.



K-means visualization

In the plot, each observation is represented by a point. The color of the points represents the cluster groups to which the observations belong. Since the plot is based on the first two components, which may not capture all the variability in the data, there might be a lot of points that overlap each other. This overlap occurs because the plot is limited to two dimensions, and some of the information from the additional variables may not be fully represented.

Hierarchical clustering methods

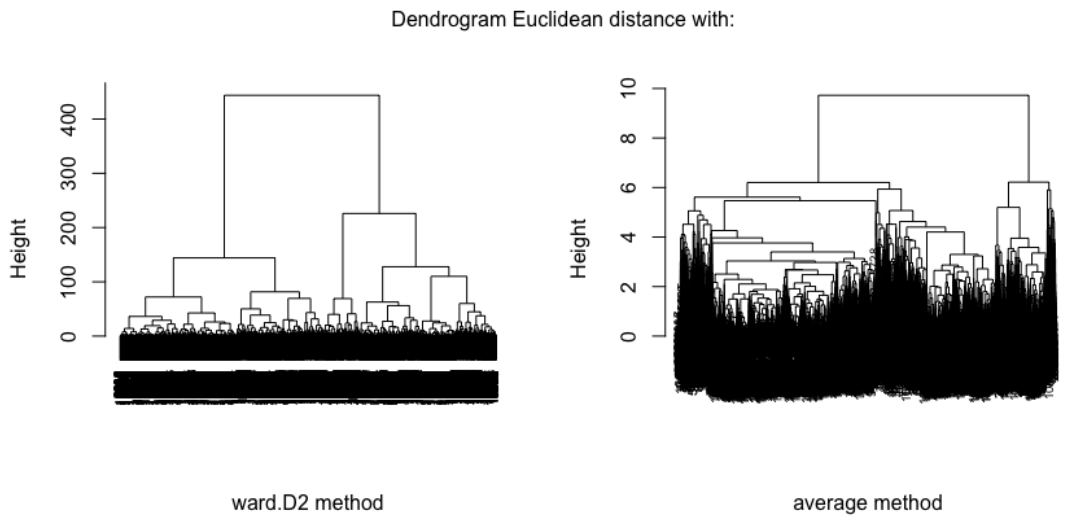
To implement the hierarchical clustering method we first used the Correlation based method (present in the r script) but the clusters are not as well defined as the values chosen, that are the euclidean distance and the method ward.D2 and average.

Distance based method

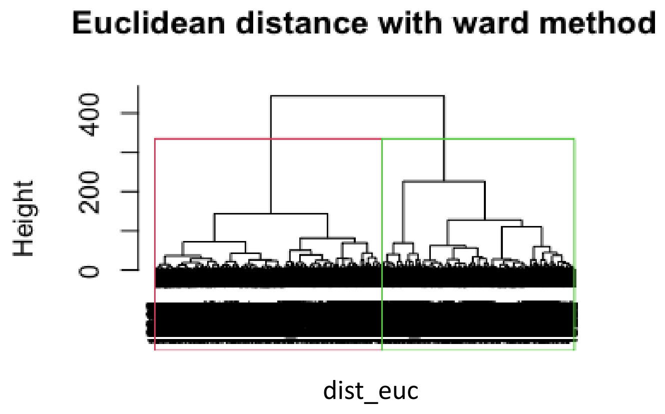
```r
# Convert all columns to numeric
sc_num_v[] <- lapply(sc_num_v, as.numeric)

dist_euc <- dist(sc_num_v, method = "euclidean")
```

```
hc_euclidean <- hclust(dist_euc, method = "ward.D2")
hc1_euclidean <- hclust(dist_euc, method = "average")
```

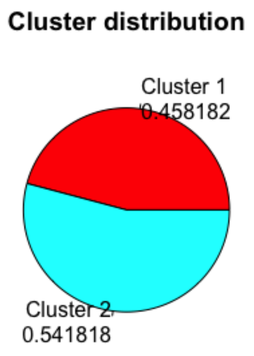
Clusters are visually represented in a hierarchical tree called a dendrogram and the choice of the ideal number of clusters is done by cutting where the height of the dendrogram is bigger.



From the plot we notice that, following the ward.D2 method dendrogram, the number of clusters is 2. Meanwhile, with the average method, the number of clusters is higher and the data are grouped in an unbalanced way. So the ward.D2 method is the most favorable one.



The colourful rectangle let us graphically understand how the data are distributed in the 2 clusters. We can already notice that the data are unbalanced but to be sure we will find the number of observations in each cluster and a graph with their proportion.



The following code adds a new column ‘cluster’ to the data frame *final_data* which now contains the cluster labels obtained from the clustering algorithm.

```
#append cluster labels to original data
final_data <- cbind(sc_num_v, cluster = groups)
```

8. Conclusions on Clustering

To draw a complete conclusion we print the proportion of point in each cluster for ward_d2 method and for k-means as well.

```
## [1] "The cluster's proportion for ward.D2 method:"

## 
##         1         2 
## 0.4581818 0.5418182

## [1] "The cluster's proportion for for k-means:"

## 
##         1         2 
## 0.6770909 0.3229091
```

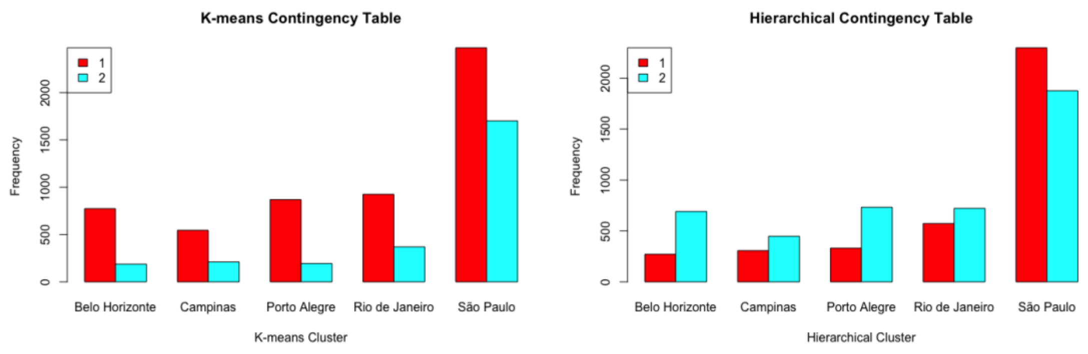
The distribution of points across the clusters from the hierarchical procedure are balanced. On the other hand from the k-means procedure, we notice that the data are unbalanced since one cluster is the double of the other one, so significantly greater.

Anyway, the clusters resulting from the hierarchical procedure exhibits a balanced distribution as all the groups have a proportion of points between .45 and .55 as values

Compare clustering results

To compare the clustering results with the city variable, we can use a contingency table. This will allow us to observe the distribution of cities within each cluster and assess any potential relationship between the clusters and the cities. We first create the contingency table using the kmean result and then the hierarchical result.

```
sc_num_v$city <- finaldata$city
# Create the contingency table with k-means
cont_table_kmeans <- table(km_1$cluster, sc_num_v$city)

# Create the contingency table with hierarchical clustering
cont_table_hierarchical <- table(final_data$cluster, sc_num_v$city)

# Set up the plotting layout with two plots side by side
par(mfrow = c(1, 2))

# Plot the barplot for k-means
barplot(cont_table_kmeans, beside = TRUE, legend = rownames(cont_table_kmeans),
        args.legend = list(x = "topleft"),
        main = "K-means Contingency Table",
        xlab = "K-means Cluster", ylab = "Frequency", col = rainbow(2))

# Plot the barplot for hierarchical clustering
barplot(cont_table_hierarchical, beside = TRUE, legend =
rownames(cont_table_hierarchical),
        args.legend = list(x = "topleft"),
        main = "Hierarchical Contingency Table",
        xlab = "Hierarchical Cluster", ylab = "Frequency", col = rainbow(2))
```



We calculated and displayed the contingency tables for the cluster labels obtained from the k-means and hierarchical clustering results, respectively. The contingency tables provide insights into the distribution of cluster labels across different cities, we can see that the predominance of the houses can be found in San Paulo for both the clusters in the two algorithms.

Adjusted Rand Index

We can compare the agreement between different clustering partitions using a variety of methods. An useful one is the Adjusted Rand Index, which is defined between -1 (perfect disagreement) and 1 (perfect agreement), and has expected value equal to zero in the case of random partition.

```
## [1] "The adjusted Rand index for the k_means is:"

## [1] -0.02576698

## [1] "The adjusted Rand index for the ward.D2 method is:"

## [1] 0.0117441
```

The adjusted Rand index values are close to 0, so it indicates that the clustering methods produce clusters that are independent or unrelated to each other. In this case the two partitions, the clusters found and the cities are independent. The reference clustering (cities) does not represent well our clusters. It confirms the reason of the asymmetry of our data and as a matter of fact it demonstrates that inside the same cluster there is no correlation with the cities.